Amazon Electronics Recommendation System

Wei Jiang
Michigan State University
East Lansing
Michigan
jiangw14@msu.edu

Shun Ran
Michigan State University
East Lansing
Michigan
ranshun@msu.edu

ABSTRACT
This project is named AERS (Amazon Electronics Recommendation System). AERS predicts a list of top 3 recommendations based on 6 algorithms (SVD, NMF, Baseline, KNN-user-based, KNN-item-based, Random) for every user. A web-based interface is also provided to allow an existing user to login. Source code and trained models are open sourced. Based on the model evaluation, SVD has best performance in terms of RMSE and MAE. The datasets are Amazon electronics reviews and the product metadata from May 1996 – July 2014 [1].

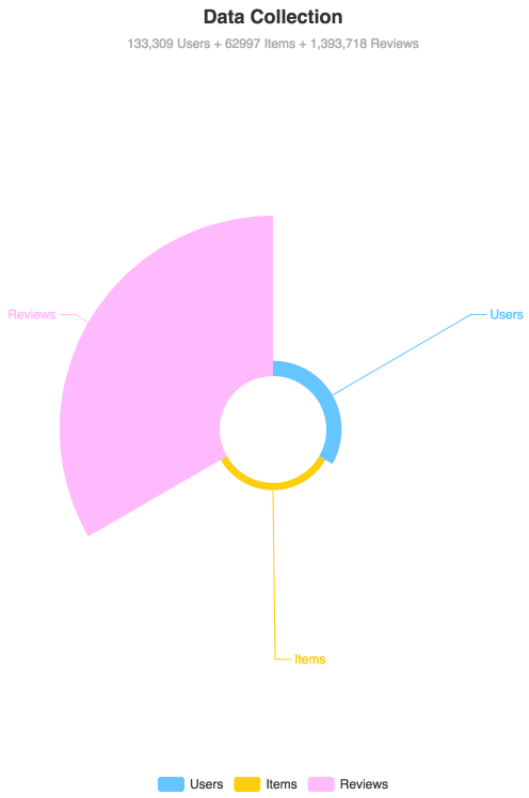
Keywords
AERS, SVD, NMF, Baseline, KNN-user-based, KNN-item-based, RMSE, MAE, Collaborative filtering, Matrix factorization

1. INTRODUCTION
Recommendation systems are widely used to address the information bottleneck problem. For example, in e-commerce websites such as Amazon, where there are millions of items/products sold, finding the right item can be a challenging task.

During previous months, we were introduced several recommendation algorithms from the Big Data course (http://www.cse.msu.edu/~ptan/CSE482/). Such as item-based or user-based nearest neighbor, principal component analysis, baseline, collaborative filtering, matrix factorization and so on. We have learned how to use some tools to do the recommendation and prediction tasks. The tools are python sklearn module, scikit-surprise, hadoop, pig, Hive, Spark and so on. We implemented a movie recommendation algorithms with different tools in several weekly exercises.

We had several challenges while undertaking such a project regarding to data collection/preprocessing, model building/evaluation and Web-based interface. As to data collection/preprocessing, neither of us had experience dealing with such a large data. In the past, the dataset for exercise was usually ~10 MB. But now the largest dataset size is ~10 GB in this project. As to the model building/evaluation, because the size issue, we had to consider the memory and time cost while doing the preprocess, tool selecting such as local VS. hadoop and model selecting such as SVD, NMF, Baseline, KNN-user-based, KNN-item-based and so on. As to the Web-based interface, we had to build the frontend and backend for AERS. We had to test out how to implement the recommendation function by choosing to make a static website or dynamic website. If dynamic, where should we host and does the server have enough memory and fast CPU to run model prediction? If static, there are will be some limitation for login which indicated that only certain small number user can log into the website.

Here are the final evaluation results:



	SVD	NMF	Baseline	KNN-user-based	KNN-item-based	Random
RMSE	1.0894	1.2499	1.0860	1.1514	1.1514	1.5026
MAE	0.8073	0.9629	0.8153	0.8909	0.8909	1.1198

2. PRELIMINARIES
Among all the 142.8 million reviews spanning May 1996 - July 2014, we only focused on Electronics and this reduced the size from 20 GB to 1.48 GB. And we filter out any user who has less

than 5 reviews. Based on the remaining user dataset, we regenerated the review data to 1.26 GB. And make the train dataset from the review dataset. We also collected the product metadata (10.54 GB) and reduce it to 12.5 MB by filtering out non-electronics and unneeded info (related, saleRank, brand, categories) in metadata.

reviews_Electronics_5.json Sample:


```
{
  "reviewerID": "A2SUAM1J3GNN3B",
  "asin": "0000013714",
  "reviewerName": "J. McDonald",
  "helpful": [2, 3],
  "reviewText": "I bought this for my husband who plays the piano.  He is having a wonderful time playing
  "overall": 5.0,
  "summary": "Heavenly Highway Hymns",
  "unixReviewTime": 1252800000,
  "reviewTime": "09 13, 2009"
}
```


metadata.json Sample:


```
{
  "asin": "0000031852",
  "title": "Girls Ballet Tutu Zebra Hot Pink",
  "price": 3.17,
  "imUrl": "http://ecx.images-amazon.com/images/I/51fAmVkTbyL._SY300_.jpg",
  "related":
  {
    "also_bought": ["B00JHONN1S", "B002BZX8Z6", "B00D2K1M3O", "0000031909", "B00613WDTQ", "B00D0WDS9A",
    "also_viewed": ["B002BZX8Z6", "B00JHONN1S", "B008F0SU0Y", "B00D23MC6W", "B00AFDOPDA", "B00E1YRI4C",
    "bought_together": ["B002BZX8Z6"]
  },
  "salesRank": {"Toys & Games": 211836},
  "brand": "Coxlures",
  "categories": [["Sports & Outdoors", "Other Sports", "Dance"]]
}
```


The train dataset is generated from review dataset. It is (#userID, #itemID, #rating) format csv file.

The training models are SVD, NMF, Baseline, KNN-user-based, KNN-item-based. SVD and NMF are matrix factorization category models. They are superior to classic nearest neighbor techniques for producing product recommendations, allowing the incorporation of additional information such as implicit feedback, temporal effects, and confidence levels. Baseline, KNN-user-based and KNN-item-based are collaborative filtering models. They are making automatic predictions (filtering) about the interests of a user by collecting preferences or taste information from many users (collaborating). In another word, they are calculated by similarities.

3. METHODOLOGY

(More details): http://webdev.cse.msu.edu/~jiangw14/cse482-AERS/report.php

(reviews_Electronics_5.json [1.48GB] => userID.data [2 MB]): filter the user has more than 5 reviews

(userID.data [2 MB], reviews_Electronics_5.json [1.48 GB] => review.json.data [1.26]): reduce reviews_Electronics_5.json to reviews.json.data based on the userID.data

(reviews.json.data [1.26GB] => train.data [41.4MB]): build a [userid, itemid, rating] format csv file from reviews.json.data

(metadata.json [10.54 GB] => productID.data[693 KB], productInfo.data[12.5 MB]): filter out non-electronics and unneeded info (related, saleRank, brand, categories) in metadata.

We trained our train.data (The [userid, itemId, Rating] csv file) by 6 algorithms (SVD, NMF, Baseline, KNN-user-based, KNN-item-based, Random) under python scikit-surprise module. And then we saved each trained model. When calculating the top 3 recommendations, firstly we make all the combination with user and this user's unpurchased item. And then use this dataset with the models to predict all the combinations. Then sort the list to return the top 3 recommendation with the predicted ratings.

When training KNN-user-based and KNN-item-based, we used k=10 and pearson_baseline similarities to train the data. However due the memory issue, we cut the train size be 1% of original dataset.

The web interface will allow an existing user to login. And this user can view his history purchased item, details and reviews of the item and a list of top 3 recommendations for each algorithm. The reason that We provide the history purchase is that it is easy to visualize the accuracy of the recommendation by comparing the recommendation and history purchased items. We also post a report with all the procedures we did and visualization of the data and evaluation result.

4. EXPERIMENTAL EVALUATION

We did cross validation with folder=5 to the whole dataset for all algorithms except the KNN-user-based and KNN-item-based.

4.1 Experimental Setup

It took about 10 days to do all the preprocess. Especially when preprocessing the metadata.json (10.54 GB), it was not a strictly valid json file. We tried to use regex and other ways to retrieve the data. But by this way, we missed some data for product. After several days' trying, we found a way online to convert this invalid json file to valid json file. And then we continued to do the parse.

For model building, besides the time we spent to train all 6 models, we also spent almost two weeks on KNN-user-based and KNN-item-based training. We thought there was a bug in our code and spent some time debug. Finally, we figured out it is the memory issue only when running KNN in python scikit-surprise module. We also tried upload to Hadoop/Spark and implemented our own KNN algorithm, but it took very long to upload the file since the file is too big. We changed our plan to use scikit-surprise KNN to train only small set of datasets. So only 1% of all dataset is being trained in KNN-user-based and KNN-item-based.

For web interface, it took roughly three weeks to do all the frontend and backend. Initially we tried to implement the website dynamically which means we will run the predict script along with all the models real time. But black.cse.msu.edu server would not let student install custom python package. So, we had to change our framework in web app, and made it load statically which means load all the prediction from the pre-generated prediction dataset.

We used cross validation with folder=5 to evaluate all the models by gathering the RMSE for test set, MAE for test set, Fit time and Test time.

Preprocessing, model building and evaluation are finished on our own MacBook with Intel CORE 5 and 8 GB RAM. And the web interface is hosted on cse.msu.edu server with Linux system.

The software we used is python scikit-surprise package which is made for building and analyzing recommender systems.

Timeline:

Date	Task	Name
02-12-2018	project proposal	Shun Ran
02-18-2018	Sent email to the owner of the dataset to request data.	Shun Ran
02-26-2018	Got valid link from owner and downloaded raw data.	Shun Ran
03-10-2018	Finish all Data preprocess and Data collection.	Wei Jiang
03-11-2018	Start design and create the layout for web interface	Wei Jiang
03-15-2018	Build Random model	Shun Ran
03-15-2018	Build Baseline model	Shun Ran
03-18-2018	intermediate report	Shun Ran
03-22-2018	Build SVD model	Wei Jiang
03-22-2018	Build NMF model	Wei Jiang
04-02-2018	Build KNN-item-based model	Wei Jiang
04-02-2018	Build KNN-item-based model	Wei Jiang
04-10-2018	Finish 80% frontend	Shun Ran
04-12-2018	Implement login in web interface	Shun Ran
04-22-2018	Finish evaluations for all models	Wei Jiang
04-24-2018	Finish web interface history and recommendation page view	Wei Jiang
04-28-2018	Finish web interface report page view	Wei Jiang
04-29-2018	Finish the report and submit project	Shun Ran

4.2 Experimental Results

Project url:

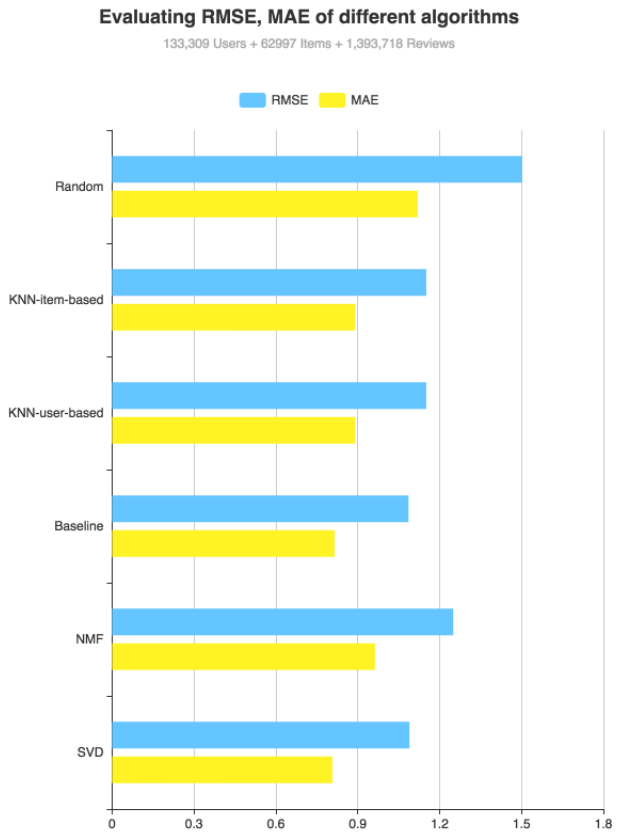
http://webdev.cse.msu.edu/~jiangw14/cse482-AERS/

Project Visualization Report:

http://webdev.cse.msu.edu/~jiangw14/cse482-AERS/report.php

Project Github address:

https://github.com/by-the-w3i/AERS



Model Evaluation

Evaluating RMSE, MAE of algorithm SVD on 5 split(s).

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	Std
RMSE (testset)	1.0882	1.0904	1.0861	1.0923	1.0900	1.0894	0.0021
MAE (testset)	0.8063	0.8085	0.8051	0.8094	0.8073	0.8073	0.0015
Fit time	83.23	83.74	83.03	83.52	83.95	83.49	0.33
Test time	4.13	3.51	3.93	3.80	3.49	3.77	0.24

Model Evaluation

Evaluating RMSE, MAE of algorithm NMF on 5 split(s).

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	Std
RMSE (testset)	1.2482	1.2500	1.2514	1.2519	1.2480	1.2499	0.0016
MAE (testset)	0.9617	0.9628	0.9638	0.9642	0.9620	0.9629	0.0010
Fit time	533.69	516.64	517.18	518.09	515.83	520.29	6.74
Test time	3.41	2.99	3.44	3.00	3.01	3.17	0.21

Model Evaluation

Evaluating RMSE, MAE of algorithm BaselineOnly on 5 split(s).

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	Std
RMSE (testset)	1.0857	1.0853	1.0880	1.0856	1.0852	1.0860	0.0011
MAE (testset)	0.8155	0.8140	0.8161	0.8155	0.8155	0.8153	0.0007
Fit time	5.12	5.40	5.23	5.29	5.22	5.25	0.09
Test time	2.97	3.33	2.62	3.01	2.97	2.98	0.22

Model Evaluation

Evaluating RMSE, MAE of algorithm KNN-item-based on 1% data as trasinset and another 1% data as testset.

RMSE	MAE
1.1514	0.8909

Model Evaluation

Evaluating RMSE, MAE of algorithm KNN-user-based on 1% data as trasinset and another 1% data as testset.

RMSE	MAE
1.1514	0.8909

Model Evaluation

Evaluating RMSE, MAE of algorithm NormalPredictor on 5 split(s).

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	Std
RMSE (testset)	1.5010	1.5026	1.5038	1.5047	1.5012	1.5026	0.0015
MAE (testset)	1.1187	1.1193	1.1209	1.1217	1.1183	1.1198	0.0013
Fit time	2.57	2.55	2.55	2.75	2.58	2.60	0.08
Test time	4.19	3.03	3.73	3.75	4.16	3.77	0.42

4.3 Discussion

From the result, SVD has lowest RMSE and MAE (highest accuracy). Baseline, NMF, KNN-user-based and KNN-item-based are very close but Baseline is slightly better than the others. The KNN-user-based and KNN-item-based are almost the same may because of the small train size. Even though KNN-user-based and KNN-item-based are trained from small size, but the accuracy is still better than NMF. And it is noticeable that Random has lowest accuracy. Overall, all the evaluation result looks normal.

5. CONCLUSIONS

The prediction result from models are ideal and reasonable because the distinguished difference between Random accuracy and the other models. Overall matrix-factorization algorithm (SVD, NMF) is performing better than the collaborative filtering algorithm (user-based, item-based). As pointed out above, the KNN-user-based and KNN-item-based may affected by the small size. In the future, if there is a better and lighter way to implement KNN on data, the KNN accuracy should be higher.

6. REFERENCES

[1] http://jmcauley.ucsd.edu/data/amazon/

[2] http://surpriselib.com/

[2] R. He, J. McAuley. Modeling the visual evolution of fashion trends with one-class collaborative filtering. WWW, 2016

[3] J. McAuley, C. Targett, J. Shi, A. van den Hengel. Image-based recommendations on styles and substitutes. SIGIR, 2015